VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální praxe

Individual professional practice in the company

2016 Veronika Uhrová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Veronika Uhrová**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: KVADOS, a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radoslav Fasuga, Ph.D.**

Konzultant bakalářské práce: Ing. Antonín Vaněček

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

V Ostrave, dňa 12.4.2016 .....

Súhlasím so zverejnením tejto bakalárskej práce podľa požiadavkov čl. 26, odst. 9 Študijného a skúšobného rádu pre štúdium v bakalárskych programoch VŠB-TU Ostrava.

KVADOS®
SOFTWARE SOLUTIONS
5. KVADOS, a.s.
Pivovarská 4710, 702 00 Ostrava 1
IČ: 25826654, DIC: CZ25826654

V Ostrave, dňa 12.4.2016 .....

Moje poďakovanie patrí spoločnosti KVADOS, a.s., ktorá mi umožnila vykonávať odbornú prax a predovšetkým Ing. Antonínovi Vaněčkovi za pomoc a ochotu pri riešení úloh. Ďalej ďakujem Ing. Radoslavovi Fasugovi, PhD. za odborné konzultácie a vedenie mojej bakalárskej práce.

Abstrakt

Táto bakalárska práca popisuje moju odbornú prax v spoločnosti Kvados, a.s. Najskôr je predstavená spoločnosť Kvados, a.s. a moje pracovné zaradenie. Ďalej sú popísané úlohy, ktoré som v rámci individuálnej praxe vykonávala a technológie a postupy, ktoré som využívala pri riešení týchto úloh. V záverečnom súhrne popisujem znalosti, ktoré som nadobudla popri štúdiu a využila na praxi, ale aj tie, ktoré mi počas praxe chýbali. Na záver zhodnocujem dosiahnuté výsledky.

Klíčová slova: Kvados, a.s., informačný systém, .NET, C#, OWIN, Nancy, trojvrstvová architektúra, kontajner, Azure, Docker

Abstract

This thesis describes my individual professional practise in the company Kvados, a.s. First of all, I describe the company Kvados, a.s. and my position in this company. Then, I describe my tasks in the practice and all technologies, that I used in implementation. In the end I describe my knowledge and skills, that I earn in school and used in practice and knowledge and skills that I during the practice missed, too. In the end there are also my results and its overall evaluation.

Key Words: Kvados, a.s., information system, .NET, C#, OWIN, Nancy, three-layer architecture, container, Azure, Docker

Obsah

Seznam použitých zkratek a symbolů

ISO	–	International Organization for Standardization
OWIN	–	Open Web Interface for .NET
API	–	Application Programming Interface
IIS	–	Internet Information Services
IoC	–	Inversion of Control
EF	–	Entity Framework
CSS	–	Cascading Style Sheets
DSL	–	Domain Specific Language
HTML	–	HyperText Markup Language
OOP	–	Object-oriented programming
VM	–	Virtual Machine
OS	–	Operačný systém
NAT	–	Network Address Translation
CLR	–	Common Language Runtime
WCF	–	Windows Communication Foundation
DHCP	–	Dynamic Host Configuration Protocol

Seznam obrázků

Seznam tabulek

1 Úvod

Praktická aplikácia poznatkov získaných v škole je dôležitým prvkom vzdelania. Z tohto dôvodu som sa rozhodla vybrať si odbornú prax ako formu svojej bakalárskej práce. Mojim zámerom bolo hlavne nadobudnúť cenné skúsenosti s prácou vo firme.

Svoju odbornú prax som vykonávala v spoločnosti KVADOS, a.s., kde som sa stala členom tímu v oddelení výskumu a vývoja.

Behom praxe som bola postavená pred niekoľko úloh. Prevažne sa tieto úlohy týkali štúdia nových technológií, skúmaním ich možností, nachádzaním ich hraníc a dôležitých predností, ale aj ich nedostatkov.

2 Profil spoločnosti

2.1 Prvé stretnutie so spoločnosťou

O spoločnosti KVADOS, a.s. som sa dozvedela prostredníctvom zoznamu firiem poskytovaným univerzitným systémom KatIS. Zaujala ma pozícia programátora v .NET a prieskum v oblasti Docker a Windows kontajnerov. Keďže išlo o pomerne novú technológiu, bola veľká šanca prebádať a naučiť sa niečo nové.

Výberové konanie spočívalo z dvoch vstupných pohovorov. Prvý s personalistkou spoločnosti Petrou Sýkorovou, ktorá si overila základné informácie o mne, ako aj znalosť anglického jazyka a druhý pohovor s vedúcim vývojárskeho týmu Antonínom Vaněčkom a výrobným riaditeľom Jiřím Vidlářom, ktorí si overili moje znalosti dosiahnuté v priebehu štúdia.

Po rozhodnutí na základe obidvoch pohovorov mi bola poskytnutá možnosť absolvovať odbornú prax v tejto spoločnosti na pozícii stážista na oddelení výskumu a vývoja.

2.2 O spoločnosti

Spoločnosť KVADOS, a.s. sa zaoberá vývojom vlastných softwarových riešení. Zameriava sa predovšetkým na klientov zo segmentu obchodu a služieb. Týmto klientom poskytuje software, ktorého cieľom je pomôcť zlepšiť kvalitu riadenia procesov.

KVADOS, a.s. pôsobí v Ostrave už od roku 1992 a od vtedy svoje pôsobenie rozšírila až do 11 európskych krajín. Centrála spoločnosti sídli v Ostrave - Mariánskych horách a zamestnáva približne 150 ľudí.

KVADOS, a.s. si zakladá na dodržovaní medzinárodných noriem ISO a preto už obdržal radu certifikácií - management kvality, riadenie projektov, enviromentálny management, systém riadenia bezpečnosti a ochrany zdravia pri práci, poskytovanie služieb IT a riadenie informačnej bezpečnosti. Kvôli dodržovaniu ISO noriem sa firma radí medzi elitu v oblasti informačných a komunikačných technológií.

Medzi spokojných zákazníkov sa radí veľmi veľa tuzemských ale aj zahraničných spoločností. Najznámejšími klientami sú ČEZ, a.s., Kofola, Tchibo, Sazka, Česká pošta, Loreál a rada ďalších.

3 Zadané úlohy

Náplň odbornej praxe bola práca na dvoch komplexnejších úlohách. Prvou úlohou bolo vytvoriť informačný systém na platforme .NET s dosiahnutím čo najväčšej možnosti testovania a znovupoužiteľnosti kódu. Druhou úlohou bolo nasadenie vytvorenej aplikácie do kontajneru a teda výskum možností hostovania aplikácie v Docker kontajneri a Windows kontajneri.

3.1 Stručný popis systému

Cieľom bolo vyvinúť informačný systém pre rezerváciu lístkov. Tento systém má riešiť správu zákazníkov, umožňnuje vytvorenie rezervácií a objednávanie lístkov na rôzne udalosti.

3.2 Technologické požiadavky na serverovú stranu

Hlavným požiadavkom na technológiu použitú na serverovej strane aplikácie je ASP.NET Web API ako technológia pre vystavovanie služieb. Vzhľadom k tomu, že aplikácia bude hostovaná v kontajneri bolo ďalším technickým požiadavkom využitie rozhrania OWIN (Open Web Interface for .NET) pre hostovanie aplikácie mimo IIS (Internet Information Services).

3.3 Technologické požiadavky na klientskú stranu

Úlohou bolo vytvorenie dvoch klientských aplikácií.

- **Desktopová aplikácia**

  Desktopová aplikácia slúži ako administrátorská aplikácia pre správu systému. Hlavným technickým požiadavkom bolo využitie ASP .NET Web API v prostredí desktopovej aplikácie.

- **Webová aplikácia**

  Webová aplikácia slúži ako klientská aplikácia, ktorá bude dostupná pre všetky moderné webové prehliadače (IE9+, Opera, Mozzila Firefox, Google Chrome, Safari, Edge). Hlavným požiadavkom na klientskú webovú aplikáciu je dôraz na prispôsobenie funkčnosti aj na mobilných zariadeniach s využitím frameworku Bootstrap a CSS Media Queries.

3.4 Výskum Docker a Windows kontajnerov

Poslednou a najdôležitejšou úlohou bolo hostovanie aplikácie v kontajneri a výskum možností Docker a Windows kontajnerov.

3.5 Časová náročnosť jednotlivých úloh

Časová náročnosť jednotlivých častí úloh je vyobrazená v tabuľke:

Úloha	Počet človekodní
serverová časť aplikácie	15
desktopová aplikácia	5
webová aplikácia	15
výskum Docker a Windows kontajnerov	25

Tabulka 1: Časová náročnosť jednotlivých úloh

4 Riešenie zadaných úloh

4.1 Aplikácia

Keďže cieľom bolo vytvoriť informačný systém na platforme .NET s dosiahnutím čo najväčšej možnosti testovania a znovupoužiteľnosti kódu, pre tento cieľ som si vybrala programovací jazyk C# a rozhodla som sa použiť viacvrstvovú architektúru. Konkrétne trojvrstvovú architektúru, v ktorej jednotlivé vrstvy plnia samostatné úlohy a je možné ich vyvíjať, udržiavať a meniť nezávisle.

- **Dátová vrstva**

  Dátová vrstva je najnižšia vrstva modelu, zabezpečuje prácu s dátami.

- **Aplikačná vrstva**

  Aplikačná vrstva alebo aj biznis či funkčná vrstva je prostredná vrstva modelu. Zabezpečuje prenos informácií medzi prezentačnou a dátovou vrstvou.

- **Prezentačná vrstva**

  Prezentačná vrstva je vrstva užívateľského rozhrania. Je to najvyššia vrstva modelu. Je to časť, ktorá je viditeľná pre užívateľa.

4.2 Použité technológie a postupy

4.2.1 Microsoft SQL Server

Pre uloženie dát na serverovej časti som sa rozhodla pre databázový systém Microsoft SQL Server, najmä kvôli výbornej kompatibilite s technológiou .NET.

4.2.2 Entity Framework

Pre objektovo relačné mapovanie som zvolila Entity Framework, ktorý zásadne zjednodušuje prácu s databázou, sleduje zmeny jednotlivých entít a zaisťuje automatické migrácie pri zmene modelu.

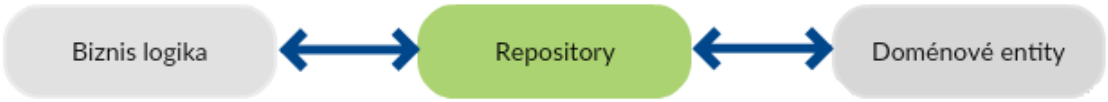
Vo svojom projekte som využila EF Code first prístup, ktorý umožňuje vytvorenie databázového modelu z doménových objektov. EF ďalej dokáže sledovať väzby medzi objektami a tieto väzby mapovať do pevného úložiska.

4.2.3 Návrhové vzory

Vo svojej práci som sa taktiež rozhodla pre využitie návrhových vzorov. Návrhové vzory môžeme definovať ako abstraktné riešenie často nachádzaných problémov z oblasti návrhu a implementácie softvéru. Predstavujú obecné riešenie problému.

4.2.3.1 Repository

Návrhový vzor Repository som využila na dátovej vrstve systému. Pracuje s doménovými entitami a vykonáva prístup k dátam. V Repository, doménové entity, logika prístupu k dátam a biznis logika spolu komunikujú na základe rozhraní. Skrýva detaily prístupu k dátam pred biznis logikou. Inými slovami, biznis logika môže pristupovať k dátovým objektom bez toho, aby vedela o architektúre uložených dát. Tým pádom v budúcnosti môžeme zmeniť architektúru alebo zdroje dát bez toho, aby to malo dopad na biznis logiku.



Obrázek 1: Návrhový vzor repository

4.2.3.2 Dependency Injection

Ďalším použitým návrhovým vzorom je Dependency Injection. Je to návrhový vzor pre predávanie závislostí medzi jednotlivými komponentami aplikácie tak, aby sa komponenty mohli vzájomne používať. Vo svojej práci som využila Constructor Injection - predávanie závislostí v konštruktore triedy.

4.2.4 IoC kontajner

Dependency injection sa používa s využitím IoC - Inversion of Control kontajnermi, ktoré v sebe držia všetky konfiguračné nastavenia týkajúce sa závislostí a riešia tieto abstrakcie na vyžiadanie. Pre technológiu .NET existuje viacero typov IoC kontajnerov. Vo svojej práci som využila Ninject, čo je jednoduchý open source IoC kontajner pre platformu .NET.

4.2.5 OWIN

Kvôli hostovaniu aplikácie v kontajneri bolo podstatným technologickým požiadavkom hostovanie aplikácie mimo IIS, takže som v mojej práci použila OWIN - Open Web Interface for .NET. OWIN definuje štandardné rozhranie medzi .NET web servermi a webovými aplikáciami.

Webová aplikácia v mojej práci je teda vytvorená ako jednoduchá konzolová aplikácia rozšírená o schopnosť hostovať webové aplikácie pomocou rozhrania OWIN. Implementáciu tohto rozhrania poskytuje NuGet balíček *Microsoft.Owin.SelfHost*.

4.2.6 Nancy framework

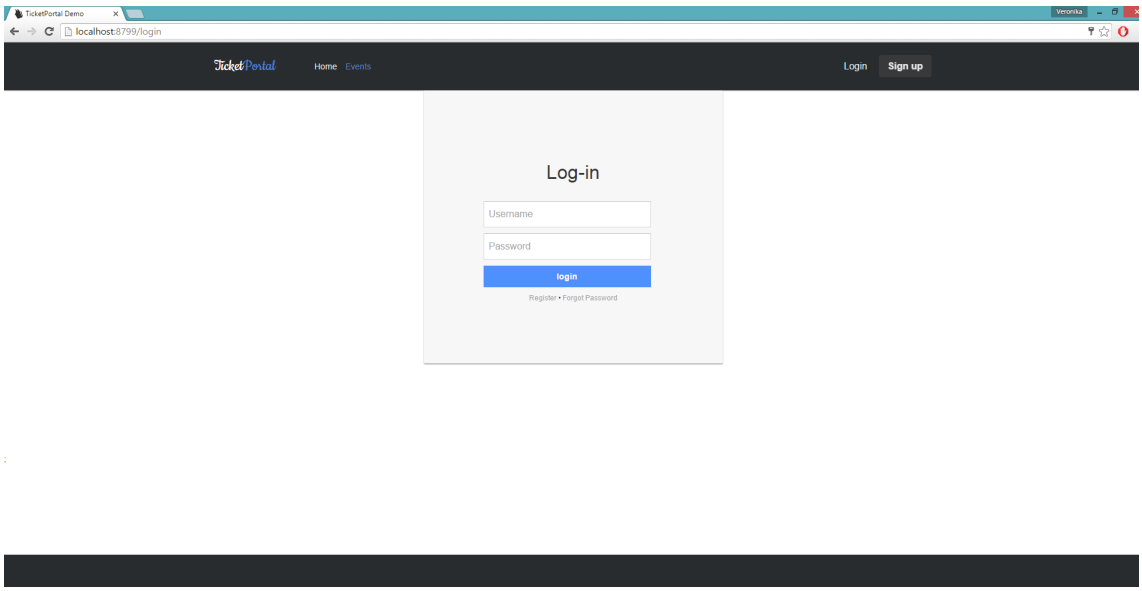
Na prezentačnej vrstve som využívala Nancy, čo je jednoduchý open-source framework pre vytváranie služieb založených na protokole HTTP pre .NET a Mono. Nancy framework bol vytvorený pre spracovávanie DELETE, GET, HEAD, OPTIONS, POST, PUT a PATH požiadavkov a prevádzanie jednoduchého DSL pre vrátenie odpovedí.

Jedným z kľúčových konceptov v Nancy je host. Host pracuje ako adaptér pre hostiteľské prostredie a Nancy, čo umožňuje Nancy bežať na technológiách ako je ASP.NET, WCF alebo OWIN.

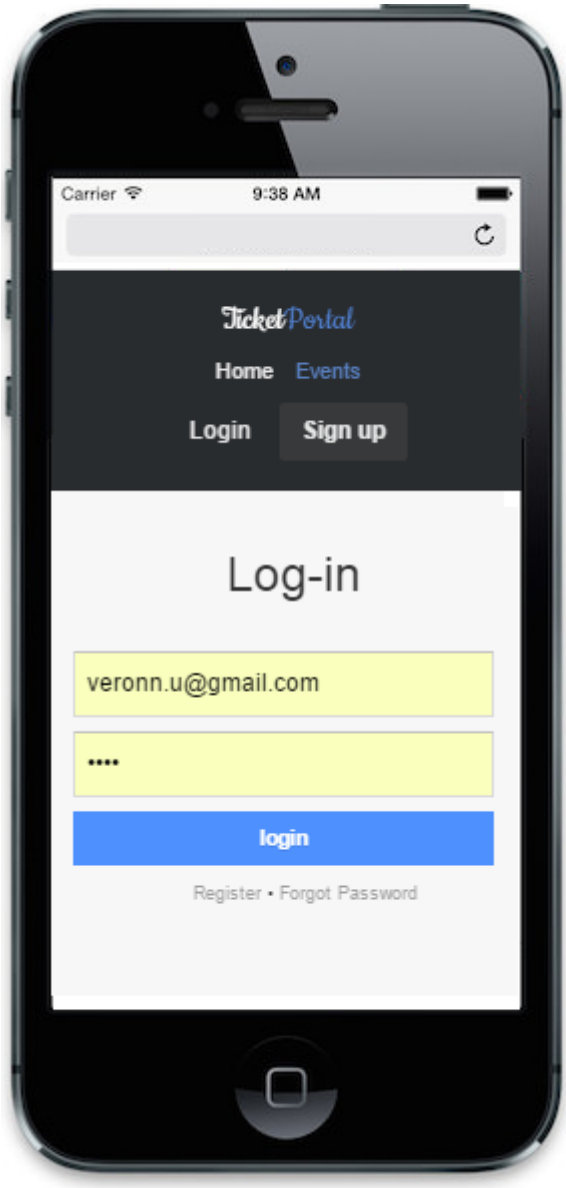
4.2.7 Bootstrap

Vzhľad aplikácie som riešila s využitím Bootstrapu a vlastných CSS štýlov. Bootstrap je voľne dostupná open-source kolekcia nástrojov pre vytváranie moderných webových stránok a webových aplikácií pomocou HTML, CSS a JavaScriptu. Bootstrap disponuje veľkou ponukou prednastavených komponent, gridov, layoutov, formulárov a rôznych prvkov, ako aj JavaScript rozšírenia.

Na vytvorenie responzívneho dizajnu aplikácie som využívala CSS media queries.



Obrázek 2: Login stránka - full screen



Obrázek 3: Login stránka - mobilné zariadenie

4.2.8 Web API

Medzi technické požiadavky na aplikáciu patrilo využitie Web API v prostredí desktopovej aplikácie. Vytvorila som ASP .NET Web API kontrolery, ktoré serializujú dáta z databáze do .json formátu a potom pomocou HTTP protokolu tieto dáta zobrazujem do desktopovej aplikácie.

4.3 Kontajnery

Mojou ďalšou komplexnou úlohou bolo nasadenie vytvorenej aplikácie do Windows kontajneru. Kontajnery sú trendom tejto doby, pretože poskytujú IT expertom pri práci s aplikáciami väčšiu

flexibilitu.

Aj keď kontajnery existujú už niekoľko desiatok rokov, spopularizované boli až v poslednej dobe s vydaním open source projektu Docker Engine.

Kontajner je izolované, prenosné operačné prostredie s odľahčenou formou virtualizácie. Zjednodušene povedané, je to balíček do ktorého je možné "zabaliť" aplikáciu aj s dátami, čo prináša veľké výhody: jednoduchú migráciu dát na iné miesto a lepšiu vnútornú kompatibilitu.

V podstate je kontajner izolovaným miestom, kde môže bežať aplikácia bez toho, aby ovplyvnila zvyšok systému a bez toho, aby systém ovplyvnil beh aplikácie.

V kontajneri beží operačný systém, má file system, a môže byť dostupný cez sieť presne tak ako fyzický alebo virtuálny počítač.

Kontajnerový prístup k správe aplikácií vedie tiež k vzostupu tzv. mikroslužieb. Pomocou nich je možné komplexné aplikácie rozdeliť na niekoľko jednotlivých komponentov, pričom každý komponent môže byť nainštalovaný v oddelenom kontajneri.

4.4 Windows kontajnery

Windows kontajnery zahŕňajú dva rozdielne typy kontajnerov:

- **Windows Server kontajnery -** poskytujú izoláciu prostredníctvom procesov a technológie izolácie menného priestoru. Windows kontajnery zdieľajú jadro s hostiteľom kontajnera a všetky kontajnery bežia na tomto hostiteľovi.
- **Hyper-V kontajnery -** rozširujú izoláciu poskytovanú Windows server kontajnermi tým, že spúšťajú každý kontajner na vysoko optimalizovanom virtuálnom stroji. V tejto konfigurácii jadro hostiteľa kontajneru nie je zdieľané s Hyper-V kontajnerom.

  Vo svojej práci som sa zameriavala na Windows Server kontajnery.

4.5 Windows server kontajnery lokálne

Pre prácu s windows kontajnermi lokálne som využila nástroj Hyper-V.

Požiadavky pre prácu s kontajnermi:

- Systém, na ktorom beží Windows 10 / Windows Server Technical Preview 4
- povolená rola Hyper-V
- 20GB dostupného úložiska pre kontajner host image, základný OS image, konfiguračné skripty
- administrátorský prístup na hostiteľskom počítači

4.5.1 Základné pojmy

Pred začatím práce s Windows server kontajnermi som si musela ozrejmiť niekoľko pojmov potrebných pri práci s nimi:

Container host - fyzický alebo virtuálny počítač.

Container image - je v podstate nemenný súbor, ktorý slúži ako predpis pre kontajner. Je to softvér, ktorý môžeme nahrať do kontajneru. Dá sa teda povedať, že kontajner je inštancia imageu.

Container OS image - je to prvá vrstva v potencionálne viac-image-vrstvovej architektúre kontajneru. Tento image poskytuje prostredie operačného systému. OS image je nemenný, nie je možné ho meniť.

Container management technology - Windows kontajnery môžu byť manažované s využitím Powershell príkazov alebo s využitím Docker enginu. S oboma nástrojmi je možné tvoriť nové kontajnery, kontajner image a tiež spravovať životný cyklus kontajnerov.

4.5.2 Vytvorenie kontajneru

V Technical Preview 4, Windows server kontajnery bežia na Windows Server 2016 a vyžadujú Windows Server 2016 Core OS Image. Pre vypísanie všetkých dostupných imageov sa využíva powershell príkaz `Get-ContainerImage`.

```
PS C:\> Get-ContainerImage
 Name                Publisher       Version     IsOSImage
 - - - -             - - - - -       - - - -     - - - -
 NanoServer          CN=Microsoft    10.0.10586  True
 WindowsServerCore   CN=Microsoft    10.0.10586  True
```

Pre vytvorenie Windows server kontajneru sa využíva `New-Container` príkaz. Nižšie uvedený výpis teda ukazuje vytvorenie kontajneru s menom TP4Demo s WindowsServerCore imageom a pripojenie kontajneru k virtuálnemu switchu s názvom Virtual Switch.

```
PS C:\> New-Container -Name TP4Demo -ContainerImageName WindowsServerCore
 -SwitchName "Virtual Switch"
 Name                          State   Uptime     ParentImageName
 - - - -                       - - - -  - - - -    - - - -
 TP4Demo                       Off     00:00:00   WindowsServerCore
```

Pre zobrazenie existujúcich kontajnerov sa používa `Get-Container` príkaz.

```
PS C:\> Get-Container
 Name        State   Uptime     ParentImageName
 ----        -----   ------     ---------------
 TP4Demo     Off     00:00:00   WindowsServerCore
```

4.5.3 Spustenie kontajneru

Kontajner sa spustí príkazom `Start-Container`.

```
PS C:\> Start-Container -Name TP4Demo
```

4.5.4 Vstup do kontajneru

Po spustení kontajneru je možné sa vstúpiť do kontajneru použitím príkazu `Enter-PSSession`.

```
PS C:\> Enter-PSSession -ContainerName TP4Demo -RunAsAdministrator
[TP4Demo]: PS C:\> C:\Windows\system32>
```

4.5.5 Výstup z kontajneru

Vystúpenie z kontajneru sa prevádza zadaním príkazu `exit`. Zadaním tohto príkazu sa vracia do hostu kontajneru.

```
[TP4Demo]: PS C:\> exit
PS C:\>
```

4.5.6 Zastavenie kontajneru

Zastavenie kontajneru sa prevádza pomocou príkazu `Stop-Container`.

```
PS C:\> Stop-Container -Name TP4Demo
[TP4Demo]: PS C:\> C:\Windows\system32>
```

4.5.7 Vytvorenie imageu z kontajneru

Vytvorené kontajnery môžu byť menené rôznymi spôsobmi. Môžu sa do nich inštalovať rôzne rozšírenia potrebné pre hostovanie aplikácie a pod. Napríklad, pomocou príkazu `Install-WindowsFeature Web-Server` môžeme do kontajneru nainštalovať IIS a následne vytvoriť z tohoto kontajneru nový image.

Postup pre vytvorenie IIS imageu:

1. Vstúpenie do kontajneru: `Enter-PSSession -ContainerName <meno-kontajneru> -RunAsAdministrator`
2. Inštalácia IIS v kontajneri: `Install-WindowsFeaute Web-Server`
3. Vystúpenie z kontajneru: `exit`
4. Zastavenie kontajneru: `Stop-Container -Name <meno-kontajneru>`
5. Vytvorenie nového imageu: `New-ContainerImage`

```
PS C:\> Enter-PSSession -ContainerName TP4Demo -RunAsAdministrator

[TP4Demo]: PS C:\> C:\Windows\system32> Install-WindowsFeature Web-Server

 Success   Restart Needed   Exit Code   Feature Result
 - - - -   - - - -          - - - -     - - - -
 True      No               Success     Common HTTP Feature, Default Document,...

[TP4Demo]: PS C:\> C:\Windows\system32> exit

PS C:\> New-ContainerImage -ContainerName TP4Demo -Name WindowsServercoreIIS -
Publisher Demo -Version 1.0.0.0

 Name                   Publisher   Version   IsOSImage
 - - - -                - - - -     - - - -   - - - -
 WindowsServerCoreIIS   CN=Demo     1.0.0.0   False
```

4.5.8 Networking

Štandardná sieťová konfigurácia pre Windows server kontajnery je vytvorenie kontajneru s pripojením k virtuálnemu prepínaču, ktorý je nakonfigurovaný ako NAT. Pre spustenie aplikácie, ktorá beží v kontajneri je nutné namapovať port kontajner hosta na port kontajneru.

V súvislosti so sieťovaním je funkcia Windows Server kontajnerov podobná funkcii virtuálnych strojov. Každý kontajner má virtuálny sieťový adaptér, ktorý je pripojený na virtuálny prepínač, cez ktorý je prevádzaná vnútorná a vonkajšia prevádzka. Sú dostupné dva typy sieťovej konfigurácie:

1. **Network Address Translation Mode -** každý kontajner je pripojený k vnútornému virtuálnemu prepínaču a bude dostávať internú IP adresu. NAT konfigurácia bude prekladať túto internú adresu na externú adresu kontajner hostu.
2. **Transparent Mode -** každý kontajner je pripojený k externému virtuálnemu prepínaču a bude dostávať IP adresu z DHCP serveru.

4.5.8.1 NAT Networking Mode

Network Address Tanslation - táto konfigurácia sa skladá z vnútorného sieťového prepínača typu NAT a WinNat. V tejto konfigurácii kontajner host má externú IP adresu, ktorá je dosiahnuteľná cez sieť. Každý kontajner má pridelenú internú IP adresu, ktorá nemôže byť prístupná cez sieť. Aby sa mohlo ku kontajneru pristupovať, externý port hostu je namapovaný na interný port, port kontajneru. Tieto spojenia sú uložené v NAT mapovacej tabuľke. Kontajner je prístupný cez IP adresu a externý port hostu, ktorý posiela prevádzku na internú IP adresu a port kontajneru. Výhodou NAT konfigurácie je, že kontajner host môže hostovať stovky kontajnerov, zatiaľ čo je použitá len jedna externe dostupná IP adresa. Ďalej, NAT konfigurácia dovoľuje viacerým kontajnerom hostovať aplikácie, ktoré môžu odpovedať identickým komunikačným portom.

- **Konfigurácia hosta**

Host konfigurácia si vyžaduje vytvorenie virtuálneho prepínača s typom NAT a nakonfigurovanie ho s vnútorným subnetom.

```
New-VMSwitch -Name "NAT"-SwitchType NAT -NATSubnetAddress "172.16.0.0/12"
```

Ďalej je potreba vytvoriť NAT objekt. Tento objekt je zodpovedný za NAT preklad adries.

```
New-NetNat -Name NAT -InternalIPInterfaceAddressPrefix "172.16.0.0/12"
```

- **Konfigurácia kontajneru**

Keď vytvárame Windows kontajner, môže byť pre kontajner vybraný virtuálny prepínač. Keď je kontajner pripojený k virtuálnemu prepínaču, ktorý je nakonfigurovaný, aby používal NAT, kontajner bude dostávať preloženú adresu.

Vytvorenie kontajneru pripojeného k virtuálnemu prepínaču, ktorý používa NAT je rovnaký ako klasické vytváranie kontajneru, iba použijeme NAT prepínač.

```
New-Container -Name DemoNAT -ContainerImageName WindowsServerCore -SwitchName "NAT"
```

- **Mapovanie portov**

Aby sa dal pristupovať k aplikáciám vnútri kontajneru, ktorý má povolený NAT je potrebné vytvoriť mapovanie portov medzi kontajnerom a kontajner hostom. Pre vytvorenie tohto mapovania, potrebujeme IP adresu kontajneru, vnútorný port kontajneru a externý port kontajner hostu.

Napríklad, pre vytvorenie mapovania medzi portom 80 kontajner hostu a portom 80 kontajneru s IP adresou 172.16.0.2:

```
Add-NetNatStaticMapping -NatName "Nat" -Protocol TCP -ExternalIPAddress 0.0.0.0 -InternalIPAddress 172.16.0.2 -InternalPort 80 -ExternalPort 80
```

Alebo napríklad pre vytvorenie mapovania medzi portom 82 kontajner hostu a portom 80 kontajneru s IP adresou 172.16.0.2:

```
Add-NetNatStaticMapping -NatName "Nat" -Protocol TCP -ExternalIPAddress 0.0.0.0 -InternalIPAddress 172.16.0.2 -InternalPort 80 -ExternalPort 82
```

Ďalej bude potrebné vytvorenie korešpondujúceho firewall pravidla pre každý externý port. Tieto pravidlá sa vytvárajú pomocou príkazu `New-NetFirewallRule`.

```
New-NetFirewallRule -Name "TCP80" -DisplayName "HTTP on TCP/80" -Protocol tcp -LocalPort 80 -Action Allow -Enabled True
```

4.5.8.2 Transparent Networking Mode

Transparent Networking - táto konfigurácia sa skladá z externého sieťového prepínača. V tejto konfigurácii každý kontajner dostane IP adresu z DHCP serveru a je dostupný na tejto IP adrese. Výhodou je, že mapovanie portov nie je potrebné.

- **Konfigurácia hosta**

Aby sa dalo nakonfigurovať kontajnerový systém, kde každý kontajner bude dostávať IP adresu z DHCP serveru, treba vytvoriť virtuálny prepínač, ktorý je pripojený k fyzickému alebo virtuálnemu sieťovému adaptéru.

Vytvorenie virtuálneho prepínača s názvom DHCP, ktorý používa sieťový adaptér pomenovaný Ethernet:

```
New-VMSwitch -Name DHCP -NetAdapterName Ethernet
```

Ak kontajner host je virtuálny stroj, musí sa povoliť `MACAddressSpoofing` na sieťovom adaptéri, ktorý sa používa s prepínačom kontajneru.

Konfigurácia na virtuálnom stroji s menom DemoVm:

```
Get-VMNetworkAdapter -VMName DemoVM | Set-VMNetworkAdapter -MacAddressSpoofing On
```

Externý virtuálny prepínač môže teraz byť pripojený ku kontajneru, ktorý je potom schopný obdržať IP adresu z DHCP serveru. V tejto konfigurácii aplikácie hostované vnútri kontajneru budú dostupné na IP adrese pridelenej kontajneru.

4.5.9 Zdieľané súbory

Zdieľané súbory vystavujú adresár z kontajner hostu do kontajneru. S vytvorením zdieľaného súboru je každý súbor v zdieľanom súbore dostupný aj v kontajneri.

Pozn.: Pre akúkoľvek prácu so zdieľanými súbormi kontajnera, musí byť daný kontajner vo vypnutom stave.

Pre vytvorenie zdieľaného súboru sa používa príkaz `Add-ContainerSharedFolder`.

```
PS C:\> Add-ContainerSharedFolder -ContainerName DEMO -SourcePath c:\datasource -
DestinationPath c:\shareddata

 ContainerName  SourcePath     DestinationPath  AccessMode
 - - -          - - -          - - -            - - -
 DEMO           c:\datasource  c:\shareddata    ReadWrite
```

Modifikovaním príkazu je možné vytvoriť zdieľané súbory napríklad len na čítanie.

```
PS C:\> Add-ContainerSharedFolder -ContainerName DEMO -SourcePath c:\source -
DestinationPath c:\dest -AccessMode ReadOnly

 ContainerName  SourcePath  DestinationPath  AccessMode
 - - -          - - -        - - -            - - -
 DEMO           c:\source    c:\dest          ReadOnly
```

Pre výpis všetkých zdieľaných súborov kontajneru sa používa `Get-ContainerSharedFolder`.

```
PS C:\> Get-ContainerSharedFolder -ContainerName DEMO2

 ContainerName  SourcePath     DestinationPath  AccessMode
 - - -          - - -          - - -            - - -
 DEMO           c:\datasource  c:\shareddata    ReadWrite
 DEMO           c:\source      c:\dest          ReadOnly
```

Pre modifikovanie už existujúceho zdieľaného súboru sa využíva `Set-ContainerSharedFolder`

```
PS C:\> Set-ContainerSharedFolder -ContainerName DEMO -SourcePath c:\source -
DestinationPath c:\dest -AccesMode ReadWrite
```

Zdieľaný súbor sa vymaže pomocou príkazu `Remove-ContainerSharedFolder`.

```
PS C:\> Remove-ContainerSharedFolder -ContainerName DEMO2 -SourcePath c:\source -
DestinationPath c:\dest
```

4.5.10 Nasadenie aplikácie do kontajneru

So všetkými nadobudnutými a popísanými znalosťami o Windows kontajneroch bolo mojou úlohou nasadiť vytvorenú aplikáciu do Windows Server kontajneru. Čo sa týka aplikácie a potrebných súborov pre jej spustenie a beh, bolo potrebné tieto súbory preniesť do kontajneru. Vo verzii Windows Server Technical Preview 4 je vytvorená funkcia pre predávanie súborov medzi kontajner hostom a kontajnerom - funkcia zdieľaných súborov.

Pre to, aby som mohla vytvoriť zdieľaný súbor medzi kontajner hostom, čo bol v mojom prípade Hyper-V virtuálny stroj, a kontajnerom, najskôr som musela presunúť súbory do kontajner hostu. To sa dalo previesť viacerými spôsobmi, z ktorých som využívala:

- **Drag and Drop**

Prenesenie súborov bolo možné aj pomocou funkcie drag and drop. Z dôvodu, že som mala vytvoriť automatický deploy aplikácie do kontajneru, sa tento spôsob prejavil ako neefektívny.

- **Stiahnutie súborov z externého úložiska**

Ďalej bolo možné si súbory s aplikáciou nazdieľať na externé úložisko, a potom pomocou `wget` tieto súbory preniesť do kontajner hostu.

```
wget -uri https://www.dropbox.com/s/b3ejas683zypyeg/app.zip?dl=1 -OutFile c:/output.zip
```

- **Kopírovanie medzi Hyper-V VM a fyzickým počítačom**

Pre mňa najefektívnejším spôsobom ako prenieť súbory bolo použitie príkazu `Copy-VMFile`, ktorý sa dá použiť aj na prenos súborov, aj na prenos celých zložiek medzi lokálnym počítačom a Hyper-V virtuálnym strojom.

Nasadenie aplikácie teda prebiehalo v niekoľkých krokoch:

1. Nakopírovanie potrebných súborov na kontajner host.
2. Vytvorenie kontajneru s bázovým imageom WindowsServerCore. Čo sa týka sieťovania, využívala som NAT networking mode, takže kontajner som vytvorila tak, aby bol pripojený na virtuálny prepínač typu NAT.

```
PS C:\Windows\system32> New-container -Name MyContainer -containerImageName WindowsServerCore -SwitchName "Virtual Switc
h"

Name        State Uptime   ParentImageName
----        ----- ------   ---------------
MyContainer Off   00:00:00 WindowsServerCore
```

Obrázek 4: Vytvorenie kontajneru

3. Vytvorenie zdieľaného súboru medzi kontajner hostom a kontajnerom.

```
PS C:\source> Add-ContainerSharedFolder -ContainerName MyContainer -SourcePath c:\source -DestinationPath c:\clientapp

ContainerName SourcePath DestinationPath AccessMode
------------- ---------- --------------- ----------
MyContainer   c:\source  c:\clientapp    ReadWrite
```

Obrázek 5: Vytvorenie zdieľaného súboru

4. Konfigurácia portov.

```
PS C:\> Add-NetNatStaticMapping -NatName "ContainerNAT" -Protocol TCP -ExternalIPAddress 0.0.0.0 -InternalIPAddress 172.
16.0.2 -InternalPort 80 -ExternalPort 80


StaticMappingID              : 0
NatName                      : ContainerNAT
Protocol                     : TCP
RemoteExternalIPAddressPrefix : 0.0.0.0/0
ExternalIPAddress            : 0.0.0.0
ExternalPort                 : 80
InternalIPAddress            : 172.16.0.2
InternalPort                 : 80
InternalRoutingDomainId      : {00000000-0000-0000-0000-000000000000}
Active                       : True
```

Obrázek 6: Vytvorenie cesty medzi interným a externým portom

5. Nastavenie firewallu.

```
PS C:\> New-NetFirewallRule -Name "HostTCP80" -DisplayName "HTTP on TCP/80" -Protocol TCP -LocalPort 80 -Action Allow -E
nabled True


Name                  : HostTCP80
DisplayName           : HTTP on TCP/80
Description           :
DisplayGroup          :
Group                 :
Enabled               : True
Profile               : Any
Platform              : {}
Direction             : Inbound
Action                : Allow
EdgeTraversalPolicy   : Block
LooseSourceMapping    : False
LocalOnlyMapping      : False
Owner                 :
PrimaryStatus         : OK
Status                : The rule was parsed successfully from the store. (65536)
EnforcementStatus     : NotApplicable
PolicyStoreSource     : PersistentStore
PolicyStoreSourceType : Local
```

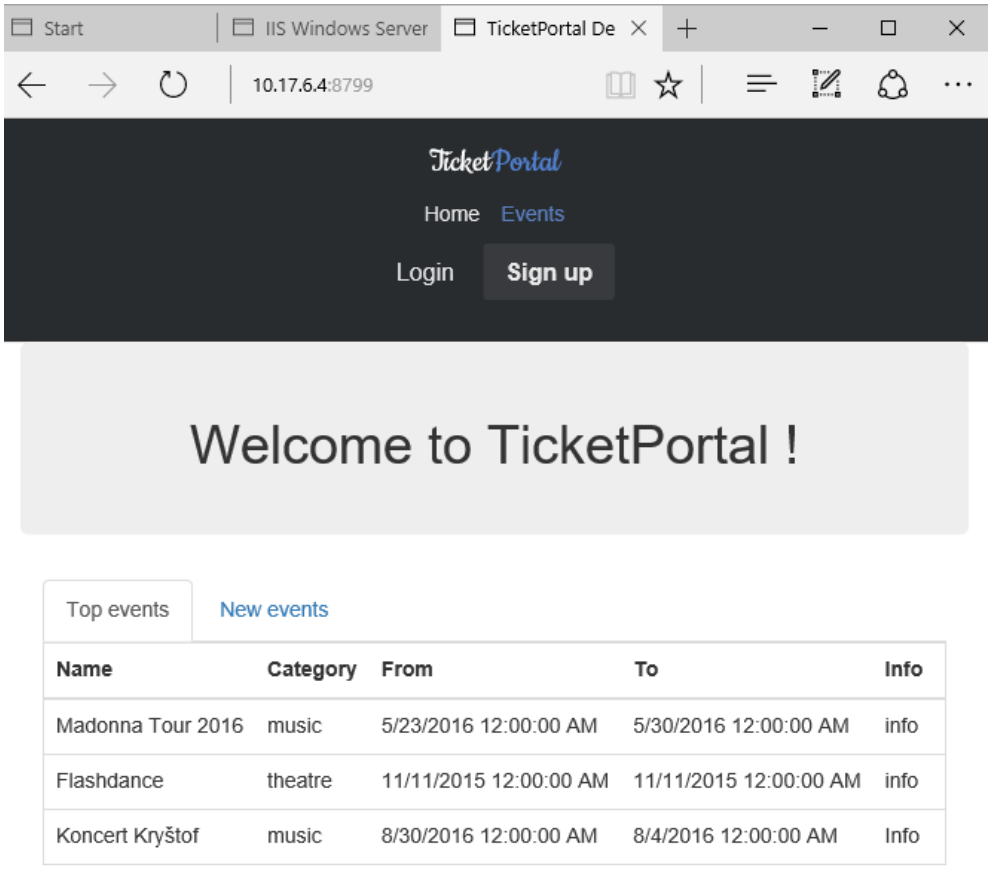
Obrázek 7: Povolenie TCP prevádzky na porte kontajner hostu

6. Spustenie aplikácie ako administrátor. Aj keď by sa zdalo, že vstúpenie do kontajneru pomocou `-RunAsAdministrator` sa užívateľ v kontajneri javí ako administrátor, opak je pravdou a aplikáciu je treba spustiť ako `Start-Process <app.exe> -Verb RunAs`.

```
[MyContainer]: PS C:\clientapp\app\appweb> start-process .\TicketPortalDemo.Web.Admin.exe -verb runas
[MyContainer]: PS C:\clientapp\app\appweb>
```

Obrázek 8: Spustenie aplikácie

Aplikáciu z kontajner hostu spustíme dotazom na IP adresu kontajneru a port aplikácie.



Obrázek 9: Aplikácia spustená vo webovom prehliadači na VM

Aplikáciu z kontajner hostu spustíme dotazom na IP adresu kontajneru a port aplikácie a z lokálneho počítača ju spustíme dotazom na IP adresu virtuálneho stroja a port aplikácie.

4.5.11 Automatické nasadenie aplikácie do kontajneru

Automatickým nasadením aplikácie do kontajneru sa myslí nasadenie aplikácie do kontajneru z lokálneho počítača. Čiže malo by byť možné po builde aplikácie z Visual Studia nasadiť aplikáciu do kontajneru a následne ju spustiť v kontajneri bez akejkoľvek ručnej konfigurácie.

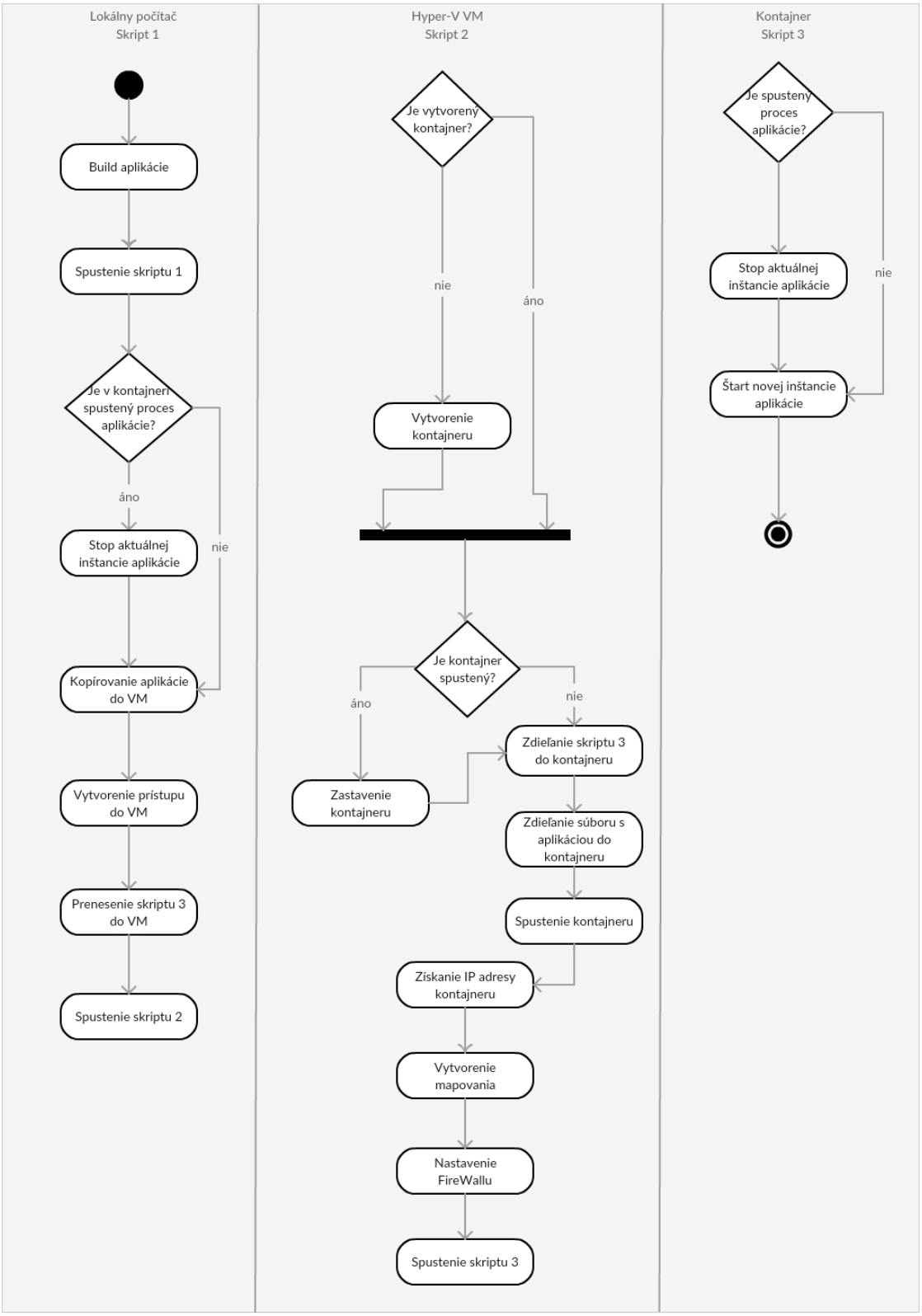
Pre tento cieľ som mienila vytvoriť powershell script, ktorý bude celú konfiguráciu riešiť. Tento úmysel však nebol možný, pretože nie je možné na lokálnom počítači spustiť príkazy určené na orchestráciu kontajnerov.

Tento skript som teda rozdelila na tri nezávislé skripty:

1. Skript pre lokálny počítač
2. Skript pre Hyper-V VM

3. Skript pre kontajner

Každý z týchto skriptov čaká na skončenie predošlého a tak je zaistené, aby sa jednotlivé príkazy vykonávali postupne v takom poradí, aby došlo k spustení aplikácie ako je potrebné.

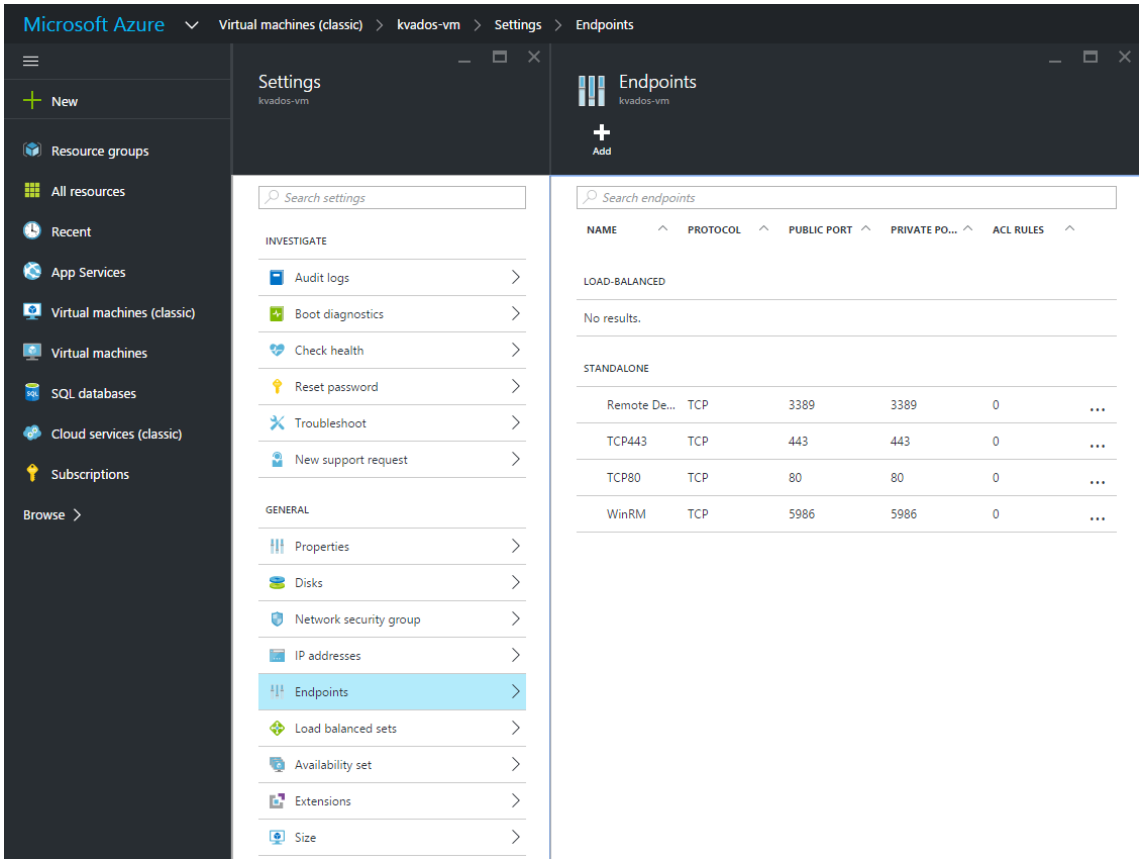


Obrázek 10: Postup automatického nasadenia aplikácie do kontajneru

4.6 Windows server kontajnery v Microsoft Azure

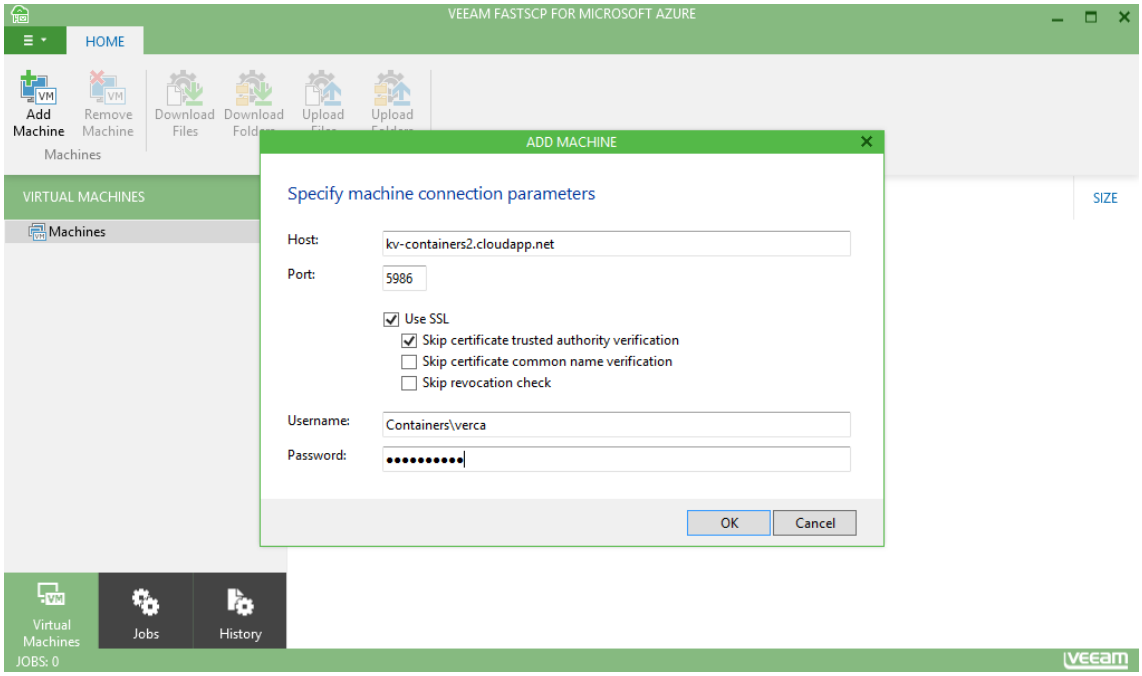
Na portáli Microsoft Azure je dostupná verzia Windows Server 2016 Core with Containers Technical Preview 4. Vytvorením virtuálneho stroja s týmto základom som vytvorila aj kontajner hosta schopného pre akúkoľvek prácu s Windows kontajnermi.

Vytvorenie virtuálneho stroja v Microsoft Azure je pomerne jednoduché. Ako prvé som virtuálnemu stroju priradila meno, užívateľa a heslo. Ďalej bolo potrebné nastaviť koncové body, ktoré boli potrebné pre ďalšiu prácu. Pre prácu s Azure VM cez vzdialenú plochu bol koncový bod už štandardne nastavený (port 3389), taktiež koncový bod pre WinRM (port 5986). Pre beh mojej aplikácie som nastavila koncové body pre protokoly HTTP a HTTPS - porty 80 a 443.



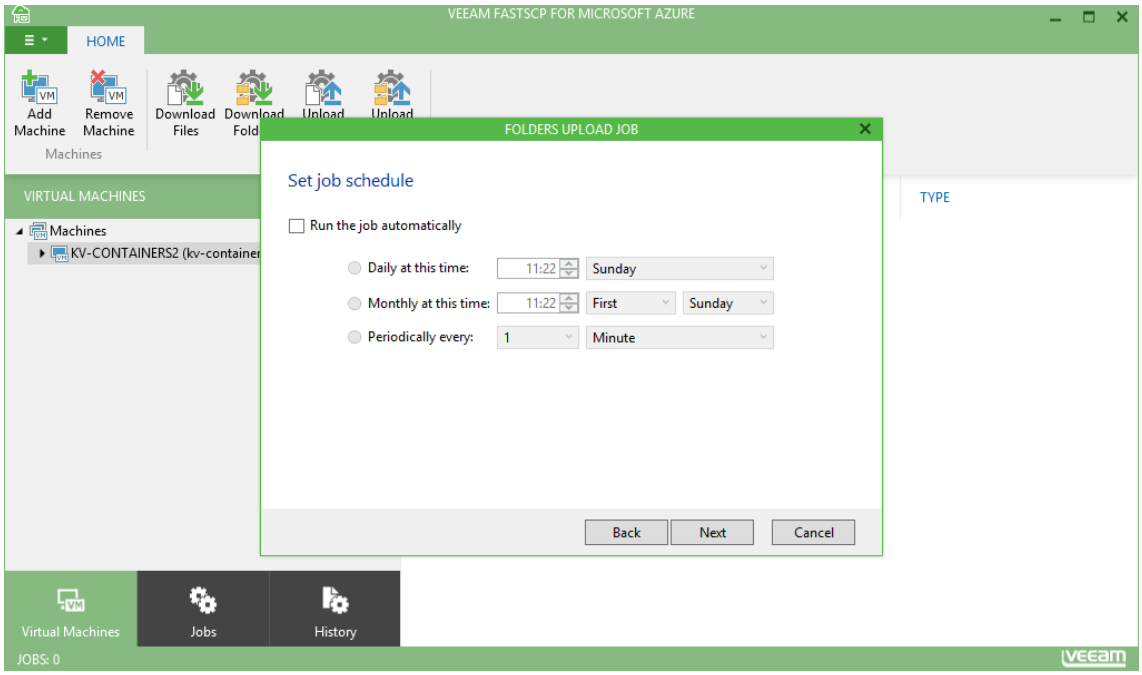
Obrázek 11: Koncové body v Azure VM

S Azure virtuálnym strojom som pracovala cez vzdialenú plochu. Nasadenie aplikácie bolo možné obdobne ako s pomocou Hyper-V virtuálneho stroja. Čo sa týka kopírovania aplikácie do kontajner hostu, do Azure VM, použila som veľmi užitočný nástroj Veeam FastSCP for Microsoft Azure. Je to nástroj, ktorý dokáže kopírovať súbory do Azure virtuálneho stroja bez potreby VPN.



Obrázek 12: Veeam FastSCP for Microsoft Azure - Pridanie VM

Nástroj naviac umožňuje automatické nasadzovanie novej verzie aplikácie do Azure virtuálneho stroja tým, že umožňuje nastavenie pravidelného kopírovania aplikácie v nastavených intervaloch.



Obrázek 13: Možnosť nastavenia pravidelných intervalov kopírovania súborov do VM

Po skopírovaní aplikácie do virtuálneho stroja, som vytvorila kontajner a zdieľaný súbor pre zdieľanie súborov uložených na virtuálnom stroji s kontajnerom.

```
PS C:\app\bin> New-Container -Name MyContainer -ContainerImageName WindowsServerCore -SwitchName "Virtual Switch"

Name        State Uptime   ParentImageName
----        ----- ------   ---------------
MyContainer Off   00:00:00 WindowsServerCore
```

Obrázek 14: Vytvorenie kontajneru v Azure VM

```
PS C:\> Add-ContainerSharedFolder -ContainerName MyContainer -SourcePath c:\app -DestinationPath c:\clientapp

ContainerName SourcePath DestinationPath AccessMode
------------- ---------- --------------- ----------
MyContainer   c:\app     c:\clientapp    ReadWrite
```

Obrázek 15: Vytvorenie zdieľaného súboru v Azure VM

Ďalej bolo potrebné pridať mapovanie na IP adresu kontajneru, nastaviť interný a externý port a povoliť firewall.

```
PS C:\> Start-Container MyContainer
PS C:\> $ipadd = Invoke-Command -ContainerName MyContainer {ipconfig | where-object {$_ -match "IPv4 Address"} | foreach
-object{$_.Split(":")[1].Trim()}}
PS C:\>
PS C:\> $ipadd
172.16.0.2
```

Obrázek 16: IP adresa kontajneru

```
PS C:\> if (!(Get-NetNatStaticMapping | where {$_.ExternalPort -eq 80})) {
>> Add-NetNatStaticMapping -NatName "ContainerNat" -Protocol TCP -ExternalIPAddress 0.0.0.0 -InternalIPAddress $ipadd -I
nternalPort 8799 -ExternalPort 80
>> }


StaticMappingID               : 0
NatName                       : ContainerNat
Protocol                      : TCP
RemoteExternalIPAddressPrefix : 0.0.0.0/0
ExternalIPAddress             : 0.0.0.0
ExternalPort                  : 80
InternalIPAddress             : 172.16.0.2
InternalPort                  : 8799
InternalRoutingDomainId       : {00000000-0000-0000-0000-000000000000}
Active                        : True
```

Obrázek 17: Nastavenie mapovania a portov

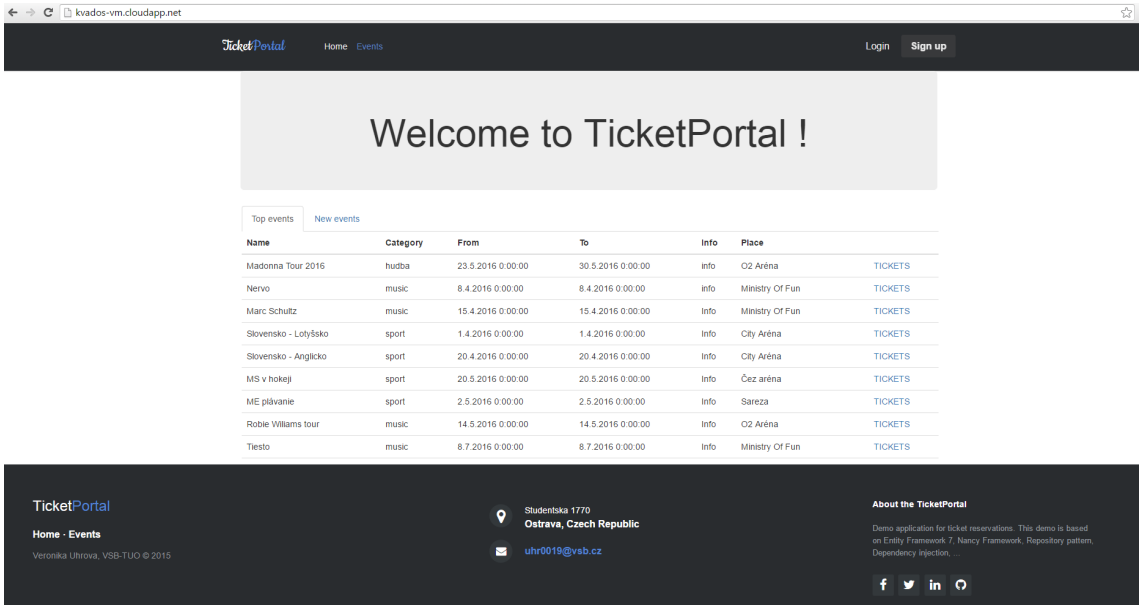
```
PS C:\> if (!(Get-NetFirewallRule | where {$_.Name -eq "TCP80"})) {
>>     New-NetFirewallRule -Name "TCP80" -DisplayName "HTTP on TCP/80" -Protocol tcp -LocalPort 80 -Action Allow -Enable
d True
>> }


Name                  : TCP80
DisplayName           : HTTP on TCP/80
Description           :
DisplayGroup          :
Group                 :
Enabled               : True
Profile               : Any
Platform              : {}
Direction             : Inbound
Action                : Allow
EdgeTraversalPolicy   : Block
LooseSourceMapping    : False
LocalOnlyMapping      : False
Owner                 :
PrimaryStatus         : OK
Status                : The rule was parsed successfully from the store. (65536)
EnforcementStatus     : NotApplicable
PolicyStoreSource     : PersistentStore
PolicyStoreSourceType : Local
```

Obrázek 18: Nastavenie firewallu pre HTTP komunikáciu

Pre spustenie aplikácie v kontajneri na Azure VM treba vojsť do kontajneru príkazom `Enter-PSSession` a spustiť aplikáciu príkazom `Start-process app.exe -Verb runas`.

Čo sa týka IP adresy, na ktorej aplikácia beží, o tú sa v tomto prípade nemusím starať, pretože Azure VM má pridelený DNS. Tým pádom sa prekladá IP adresa kontajner hosta, čiže virtuálneho stroja, na doménové meno, v mojom prípade `www.kvados-vm.cloudapp.net`.



Obrázek 19: Aplikácia spustená z kontajneru na Azure VM

4.7 Docker

Docker je open source framework, ktorý ponúka odľahčený druh virtualizácie pomocou linuxových kontajnerov.

Docker vychádza z tradičných linuxových distribúcií, ako sú Red Hat Enterprise Linux alebo Ubuntu a umožňuje zabaliť aplikácie a služby ako bitové kópie, ktoré bežia vo svojich vlastných prenositeľných kontajneroch.

Tie pritom možno presúvať medzi fyzickými, virtuálnymi a cloudovými prostrediami bez nutnosti akejkoľvek úpravy. Docker týmto spôsobom ponúka veľmi vysoký stupeň prenositeľnosti aplikácií a agility a vyhovuje tak vysoko škálovateľným aplikáciám.

Kontajner Docker je teda vo svojej podstate virtuálny stroj, ale omnoho odľahčenejší. Umožňuje napríklad hladko presúvať aplikácie a služby medzi hostiteľskými servermi. Okrem toho obsahuje nástroje pre verzovanie a správu bitových kópií, ktoré dovoľujú jednoduché škálovanie a elasticitu aplikácií a služieb pre inštancie na fyzických a virtuálnych serveroch alebo v cloude.

V októbri 2014 predstavil Docker spoluprácu so spoločnosťou Microsoft. Dovtedy sa jednalo hlavne o linuxovú platformu, Microsoft v Dockeri videl potenciál a tak sa Docker stal súčasťou Windows Server kontajnerov.

Vo Windows Server 2016 je možné Windows server kontajnery manažovať pomocou Docker príkazov. Vo Windows Server sa teda nejedná o linuxové kontajnery, ale ide o Windows kontajnery manažované Docker príkazmi.

Všetky príkazy pre manažovanie Windows kontajnerov sa začínajú kľúčovým slovom `docker` a sú diametrálne odlišné od powershell príkazov pre Windows kontajnery.

Verzia Docker enginu sa zobrazí pomocou príkazu `docker version`.

4.7.1 Vytvorenie kontajneru

Pred vytvorením kontajneru sa pomocou príkazu `docker images` zobrazia všetky image nainštalované na kontajner hoste.

```
PS C:\> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
windowsservercore	10.0.10586	6810d945fda	2 weeks ago	0 B
windowsservercore	latest	6810d945fda	2 weeks ago	0 B
nanoserver	10.0.10586	6810d945fda	2 weeks ago	0 B

Príkaz pre vytvorenie kontajneru sa môže modifikovať rôznymi tagmi.

-d - spúšťa kontajner v pozadí

-p - hovorí Dockeru aby namapoval požadované porty vnútri kontajneru na porty kontajner hostu.

V nasledujúcom príklade sa vytvorí kontajner s menom DemoContainer a základným imageom windowsservercore, spustí sa na pozadí a namapuje sa port kontajneru na port kontajner hostu.

```
PS C:\> docker run -d –name DemoContainer -p 80:80 windowsservercore

45879xd2a4d5e65a2d11
```

4.7.2 Zoznam kontajnerov

Pre výpis všetkých bežiacich kontajnerov slúži príkaz `docker ps`. Všetky kontajnery sa vypíšu modifikovaným príkazom `docker ps -all`.

4.7.3 Spustenie kontajneru

```
PS C:\> docker start DemoContainer
DemoContainer
```

4.7.4 Zastavenie kontajneru

```
PS C:\> docker stop DemoContainer
DemoContainer
```

4.7.5 Vytvorenie imageu z kontajneru

Docker tiež umožňuje vytvorenie imageu z kontajneru pomocou príkazu `commit`.

```
PS C:\> docker commit DemoContainer NewImage
4193c9f34e320c4e2c52ec52550df225b2243927ed21f014fbfff3f29474b090
```

4.7.6 Vstup do kontajneru

Ak je kontajner spustený, vstupuje sa do neho príkazom `docker exec`.

```
PS C:\> docker exec -t -i DemoContainer /bin/bash
[DemoContainer]: C:\>
```

4.7.7 Výstup z kontajneru

Výstup z kontajneru sa ako pri powershell príkazoch uskotočnuje príkazom `exit`.

```
[DemoContainer]: C:\> exit
PS C:\>
```

4.7.8 Networking

Čo sa týka sieťovania, docker príkazy neobsahujú sadu príkazov pre sieťovanie, takže sieťovanie sa konfiguruje powershell príkazmi a keďže ide o klasické windows kontajnery, ide o tie isté typy sieťovania.

4.8 Docker for Windows

Docker v spolupráci so spoločnosťou Microsoft naviac vytvorili Docker klienta, ktorý umožňuje vytvárať a spúšťať linuxové kontajnery na OS Windows.

Všetko začína inštaláciou Docker toolboxu. Docker toolbox obsahuje všetky Docker nástroje ako je Docker engine, Docker machine, Docker compose a Kinematic.

4.8.1 Docker engine

Docker engine poskytuje hlavné funkcie, ktoré sú potrebné k práci s Docker kontajnermi.

4.8.2 Docker compose

Docker compose je nástroj pre definovanie a beh multikontajnerových Docker aplikácií. S Docker compose a použitím Compose súboru je možné konfigurovať služby aplikácií. Tým pádom použitím niekoľkých príkazov sa z jednej konfigurácie vytvoria a spustia všetky služby.

4.8.3 Docker machine

Docker machine je nástroj, ktorý nainštaluje Docker engine na virtual host a môže ho spravovať pomocou `docker-machine` príkazov. Čiže nie je potrebná ručná konfigurácia virtuálneho stroja. Docker machine vie vytvoriť Docker hosta na lokálnom počítači, firemnej sieti, dátovom centre alebo cloude.

S `docker-machine` príkazmi sa dá jednoducho spustiť, vypnúť, reštartovať, manažovať host, aktualizovať Docker klienta alebo deamona a nastaviť Docker klienta na komunikáciu s hostom.

4.8.4 Kinematic

Kinematic je open source projekt vytvorený pre zjednodušenie a zefektívnenie používania Dockeru. Kinematic automatizuje inštaláciu a nastavenie Docker enginu a poskytuje intuitívne užívateľské rozhranie pre spustenie Docker kontajnerov.

4.8.5 Boot2Docker

Boot2docker je v podstate inštalátor na OSX (alebo Windows), ktorý nainštaluje Oracle VirtualBox a do neho pripraví virtuálny stroj, na ktorom bude Docker engine bežať.

4.8.6 Dockerfile

Docker dokáže vytvoriť image automaticky z takzvaného dockerfilu. Dockerfile je textový súbor, ktorý obsahuje príkazy potrebné pre vytvorenie špecifického imageu. Dockerfily dodržujú špecifický formát a používajú špecifické sady inštrukcií.

4.8.7 Nasadenie aplikácie do Docker kontajneru

Nasadenie aplikácie do Docker kontajneru pozostávalo z vytvorenia imageu s aplikáciou, následného vytvorenia kontajneru z vytvoreného imageu, nastavenia portov a nastavenia firewallu.

4.8.7.1 Vytvorenie imageu z Dockerfilu

Pre nasadenie aplikácie do Docker kontajneru je potrebné vytvoriť image, ktorý bude obsahovať aplikáciu. S pomocou tohto imageu sa potom vytvorí kontajner, ktorý bude obsahovať všetko potrebné pre beh aplikácie.

Tento image bolo možné vytvoriť s pomocou Dockerfilu. Jednoducho povedané, bolo potrebné nakopírovať súbory potrebné pre beh aplikácie do imageu.

Dockerfile pre môj aplikačný image:

```
FROM mono
COPY . /src
EXPOSE 8799
CMD ["mono","src/bin/TicketPortalDemo.Web.Admin.exe"]
```

Inštrukcia `FROM` definuje bázový image. Windowsservercore OS image som využiť nemohla, pretože Docker beží na OS Linux, pre ktorý tieto image nie sú dostupné. Použila som teda image s nainštalovaným Mono frameworkom.

Mono je open source implementácia Microsoft .NET Frameworku založený na ECMA štandardoch pre C# a CLR.

Inštrukcia `COPY` definuje do akej zložky v kontajneri sa majú súbory s aplikáciou preniesť.

`EXPOSE` definuje na akom porte v kontajneri pobeží aplikácia.

`CMD` definuje príkaz na spustenie aplikácie.

Pre vytvorenie imageu s aplikáciou slúži príkaz `docker build -t <nazov-imageu> <url-k-dockerfilu>`.

```
veronikau@uhrovav MINGW6a /c/
$     docker build -t docker-app-image:1.0.0 c:/application
Sending build context to Docker daemon 39.89 MB
Step 1: FROM mono
- - -> f8s54a5c4v
Step 2: COPY . /src
- - -> Using cahe
- - -> 1s41a4ad4w
Step 3: EXPOSE 8799
- - -> Using cahe
- - -> 1s41a4ad4w
Step 4: CMD mono src/bin/TicketPortalDemo.Web.Admin.exe
- - -> Using cahe
- - -> 1s41a4ad4w
Succesfully built 1665a0a4bf55
```

4.9 Load Balancing

V rámci výskumu Windows a docker kontajnerov bolo mojou poslednou úlohou vytvorenie load balancingu medzi dvoma alebo viacerými kontajnermi.

Load balancing alebo vyvažovanie záťaže je mechanizmus, ktorý umožňuje rozmiestniť záťaž medzi dvoma počítačmi, sieťovými pripojeniami, procesormi, pevnými diskami a teda aj kontajnermi za cieľom dosiahnuť optimálne využitie zdrojov, maximalizovanie priepustnosti, minimalizovanie času odozvy a predídeniu zaťaženia.

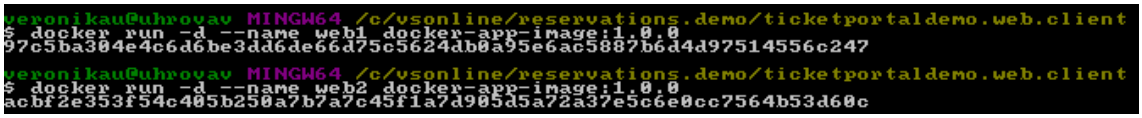
Pre vytvorenie load balancingu som sa rozhodla použiť Docker kontajnery a HAProxy.

Haproxy (High Availability Proxy) je open source softvér ktorý prevádza vysoko kvalitný load balancer a proxy server pre aplikácie založené na TCP a HTTP tým, že rozkladá žiadosti cez viaceré servere. Je napísaný v jazyku C a je používaný na viacerých vysoko profilových webových stránkach ako je GitHub, Bitbucket, Stack Overflow, Reddit a ďalších.

HAproxy je teda schopná rozdeľovať záťaž medzi aplikačnými kontajnermi. Aby toto bolo možné, potrebovala som na to image, v ktorom bola nainštalovaná aplikácia a image, v ktorom je nainštalovaná HAproxy. Image s HAproxy je voľne dostupný a štandardne beží v HTTP móde.

Vytvorenie load balancingu spočíva vo vytvorení viacerých kontajnerov a následné spojenie týchto kontajnerov kontajnerom, ktorý je vytvorený z HAproxy imageu a teda pracuje ako load balancer.

Ja som teda vytvorila 2 kontajnery z imageu s nainštalovanou aplikáciou. Oba tieto kontajnery naslúchajú na porte 8799 podľa špecifikácie v dockerile.



Obrázek 20: Vytvorenie dvoch docker kontajnerov

Následne som vytvorila tretí kontajner, v ktorom pracuje HAproxy, čiže pracuje ako load balancer, počúva na porte 80 a posiela žiadosti ku spojeným kontajnerom použitím roundrobin algoritmu.

Obrázek 21: HAproxy kontajner

5 Znalosti a zručnosti získané v priebehu štúdia a uplatnené v priebehu praxe

Pri praxi som využila veľmi veľa znalostí ohľadne OOP získaných behom štúdia. Boli to najmä znalosti z predmetu Programovacie jazyky II., kde som sa naučila využívať funkcie, ktoré platforma .NET a programovací jazyk C# ponúka.

Keďže som sa na praxi zaoberala tvorbou informačného systému, tak som využila veľmi veľa dôležitých informácií získaných na predmetoch Úvod do databázových systémov, Databázové informačné systémy a Vývoj informačných systémov. V týchto predmetoch som si osvojila základné techniky a postupy tvorby komplexných informačných systémov.

Pri tvorbe webového klienta som si osvojila znalosti z predmetu Vývoj internetových aplikácií, v rámci ktorého som si zdokonalila svoje schopnosti v tvorbe moderných webových stránok a aplikácií.

Pri práci s kontajnermi som sa stretla aj so základnou konfiguráciou sieťovania, takže som využila aj poznatky nadobudnuté v predmete Počítačové siete.

Behom štúdia som sa naučila pracovať hlavne s anglickými tutoriálmi. Túto skúsenosť som veľmi často využila na praxi, pretože som sa stretla s technológiami, s ktorými som nemala príliš mnoho skúseností.

Všeobecný prehľad, ktorý som získala behom štúdia v oblasti informačných a komunikačných technológií hodnotím veľmi kladne.

6 Chýbajúce znalosti a zručnosti v priebehu praxe

V praxi som sa snažila využiť všetky poznatky, ktoré som nadobudla štúdiom na VŠB-TU Ostrava, ale našli sa aj znalosti, ktoré mi pri práci vo firme chýbali.

Na začiatku práce mi chýbali hlbšie znalosti možností, ktoré technológia .NET ponúka. Tieto znalosti sú čiastočne súčasťou predmetu Architektúra technológie .NET, ale tento predmet som neabsolvovala.

Najväčšie problémy mi robila práca a výskum možností kontajnerov. Keďže je to nová technológia vo svete firmy Microsoft, niekedy nebolo jednoduché nájsť potrebné informácie k danej téme.

7 Záver

Vytvorený koncept som na záver svojej odbornej praxe odovzdala svojmu vedúcemu tímu k ďalšej analýze.

Absolvovanie odbornej praxe bolo pre mňa výbornou životnou skúsenosťou vo všetkých smeroch. Či už išlo o technické stránky a postupy pri implementácii pomocou programovacieho jazyka alebo medziľudskou komunikáciou vo firemnom prostredí. Behom praxe sa mi tiež potvrdilo to, že je veľký rozdiel medzi pasívnymi poznatkami a schopnosťou tieto poznatky využiť pri riešení reálneho projektu.

Veronika Uhrová

Literatura

[1] Entity Framework, *http://www.entityframeworktutorial.net/*, posledná úprava 2015

[2] Pecinovský Rudolf, *Návrhové vzory*, Computer Press, 2007

[3] Repository pattern, *http://blog.falafel.com/implement-step-step-generic-repository-pattern-c/*, posledná úprava 1. 10. 2014

[4] Trojvrstvová architektúra (Three-tier architecture) - prax, *http://www.web-integration.info/cs/blog/jak-uplatnit-principy-trivrstve-architektury-v-ramci-web-integracniho-projektu*, posledná úprava 15. 10.2013

[5] Dependency Injection, *http://www.c-sharpcorner.com/UploadFile/3d39b4/constructor-dependency-injection-pattern-implementation-in-c/*, posledná úprava 25. 4. 2014

[6] Ninject, *http://www.c-sharpcorner.com/UploadFile/4d9083/dependency-injection-using-ninject-in-net/*, posledná úprava 7. 2. 2015

[7] Owin, *http://www.c-sharpcorner.com/UploadFile/4b0136/working-with-owin-hosting-and-self-hosting-in-Asp-Net/*, posledná úprava 2. 4. 2014

[8] Nancy framework, *http://blog.bekijkhet.com/2012/04/c-nancy-selfhosting-webapplication-with.html*, posledná úprava 2. 4. 2012

[9] Bootstrap, *http://www.tutorialspoint.com/bootstrap/*, posledná úprava 2016

[10] Web API, *http://www.developer.com/net/asp/consuming-an-asp.net-web-api-using-httpclient.html*, posledná úprava 17. 5. 2012

[11] Kontajnery, *https://azure.microsoft.com/en-us/blog/containers-docker-windows-and-trends/*, posledná úprava 17. 5. 2015

[12] Windows kontajnery, *http://www.thomasmaurer.ch/2015/09/first-steps-with-windows-containers/*, posledná úprava 15. 9. 2015

[13] Windows server kontajnery, *http://msdn.microsoft.com/en-us/virtualization/windowscontainers/quick-start/inplace-setup*, posledná úprava 18. 2. 2016

[14] Docker, *https://www.docker.com/what-docker*, posledná úprava 2016

[15] Docker kontajnery, *https://technology.amis.nl/2015/03/15/my-first-steps-with-docker-starting-from-windows-as-the-host/*, posledná úprava 15. 3. 2015

[16] Load balancing, *http://blog.tutum.co/2015/05/05/load-balancing-the-missing-piece-of-the-container-world/*, posledná úprava 5. 5.2015

8 Prílohy

Súčasťou bakalárskej práce je DVD, ktorého súčasťou je:

- Zdrojový kód aplikácie
- Powershell skripty